A4: GRAPHS!

"If you're walking down the right path and you're willing to keep walking, eventually you'll make progress." — Barack Obama

Course: CS 5006
Summer 2018
Due: 8 June 2018, 5pm

OBJECTIVES

After you complete this assignment, you will be comfortable with:

- Graph traversals
- Djikstra's Algorithm
- Bipartite
- Topological ordering
- Graph colorings

RELEVANT READING

- Kleinberg and Tardos, Chapter 5

AND NOW ON TO THE FUN STUFF.

Problem 1: General graphs (10 points)

(a) (4 points) Given an adjacency-list representation of a directed graph, how long does it take to compute the out-degree of every vertex? How long does it take to compute the in-degrees? Give a short description or explanation for each.

It would take O(1) to computer the out-degree of every vertex. All you'd have to do is figure out how long the array is each vertex being tracked. The in-degree will take O(mn). In order to figure out the in degree, you have to figure out which nodes are connected to the node. For that, you potentially have to go through the full array for every other node that isn't the node you are trying to figure out the in degree for. Going through an array with be O(m), with representing the number of elements in the array for a node, which can be, but may not be, the same for each node, and m ≤ n. And you are doing this for every node that isn't the node you're trying to figure this out for. So the time to do this n-1 times. So the time would be $(n-1) * O(m) = \ O(mn) - O(m) = O(mn)$

(b) (6 points) Give an adjacency-list representation for a complete binary tree on 7 vertices. Then, give an equivalent adjacency-matrix representation. Assume that vertices are numbered from 1 to 7, starting at the top and numbering the nodes left to right at each level.

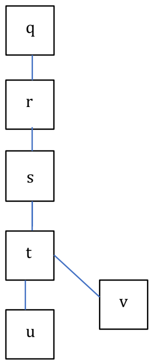
1	2	3		
2	1	4	5	
3	1	6	7	
4	2			
5	2			
6	3			
7	3			

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Assuming this is undirected

Problem 2: DFS (3 points)

Assume a graph *G* that contains a path from *u* to *v*, and *u.depth* < *v.depth* (that is, the depth of *u* is less than the depth of *v*) in a depth-first search of *G*. I propose that *v* is a descendant of *u* in the depth-first forest produced. Provide a counter-example.



Problem 3: BFS (4 points)

In class, we primarily focused on traversals using an adjacency list. Using the basic traversal algorithm, what is the running time of BFS if the graph is represented by an adjacency matrix instead? Assume the traversal algorithm is modified as necessary to handle the matrix rather than the lists.

If the graph is represented by an adjacency matrix instead: So we need to make an *nxn* maxtrix. Root node, you'd go through each index of the array and mark a 0 if the node is connected to the root and a 1 if it isn't connected to the root. This would take O(n) time. Then you'd create/go to the arrays you created that represent the arrays that were connected to the root, and repeat the same thing, with the added caveat that anything already added would automatically be marked 0, shrinking how much you'd have to traverse the each new array you were creating. You'd have to create the amount of arrays equal to the total amount of elements so the O(n) for creating stays constant, but O(n) for traversing the array shrinks as you continue down the graph so while you are doing this O(n) times, the amount of time you spend doing it is O(m) so the time is O(mn).

Problem 4: White Hats, Black Hats (6 points)

In the world of politics, there are two kinds of politicians: "white hats" (good guys) and "black hats" (bad guys). Let's assume that we don't really care about which party a politician is a part of for now. But, between any pair of politicians, there may or may not debate. Suppose we have a list of n politicians, and a list of r pairs of politicians that have a debate. Give an algorithm that determines whether it is possible to specify some politicians as white hat, and the rest as black hat, such that each debate is between a white hat and a black hat.

If it is possible to do this determination, your algorithm should produce it. The algorithm should run in $O(n + r)$ time.

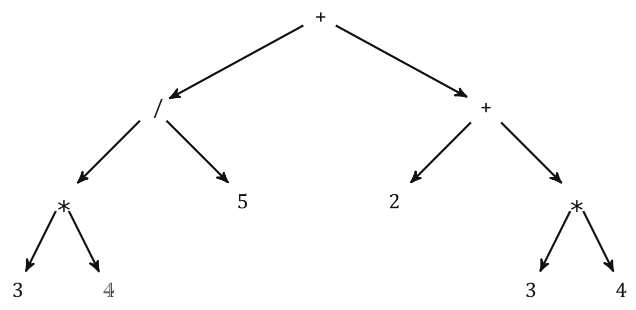
I'm not sure it's possible to create such an algorithm without more information. If n is a list of politicians, without information amount the proportions of each politicians in the list, all r tells us is the amount of politicians where we actually know whether they are WH or BH. So $2r$ would be the amount of politicians where we know for sure what they are. $n - 2r$ is the amount we don't know what they are. Unless we are assuming that the only WH politicians are those in the pair, there is no way of knowing how many remaining politicians there are who could be who are WH without knowing the proportions of n between WH and BH.

```
DeterminePoliticians(n, r):
        Initialize BHAmount = 0;
        Initialize WHAmount = 0;
        While r ≠ 0{
                BHAmount ± 1;
                WHAmount ± 1;
                r-= r-1;
        }

        AmountOfUnknownPoliticians = n-BHAmount-WHAmount;
        If (AmountOfUnknownPoliticians >0){
                Return false;
        Else{
                Return true;
        }
```

Problem 5: Graph Arithmetic (4 points)

Consider representing an arithmetic expression as a tree. Each leaf is a number (integer), and each internal node is an arithmetical operations (+, −, *, /). For example, the expression 3 * 4/5 + 2 + (3 * 4) is represented by the following tree:



Give an $O(n)$ algorithm for evaluating such an expression, where there are n nodes in the tree. In your solution, provide a 1-2 sentence summary of your algorithm, and then provide psuedocode for your algorithm.

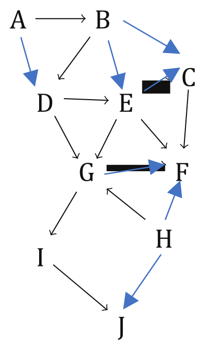
So you need to perform a depth first search with the pattern being left, middle, right, to make sure that all the operations are done in the proper order. As you are doing the depth first search, you need to store the value you are applying all the operations to until the end, otherwise you won't be able to return anything.

```
Arithmetic(root){
        Initialize B = 0;
        While (root ≠ null){
                Arithmetic(root->left);
                B = root-left, operation that's in the middle, root->right;
                Arithmetic(root->right);
                Retain value of B for next level up as new value of root-left;
        }

        Return B;
}
```

Problem 6: Topological ordering (5 points)

Do a topological sort on this graph. Please show the process as well as a valid final output.



The only one I can see that doesn't have any incoming arrows is A, so A is first:

A

After I remove A and it's lines, B doesn't have any incoming lines, so B is next:

A, B

Remove all the lines from B, and D is the one that doesn't have incoming lines:

A, B, D

Remove the lines coming from D, E doesn't have incoming lines:

A, B, D, E

Remove the lines coming from E, C doesn't have incoming lines:

A, B, D, E, C

Remove lines coming from C, H doesn't have any incoming lines:

A, B, D, E, C, H

Remove lines coming from H, G doesn't have any incoming lines:

A, B, D, E, C, H, G

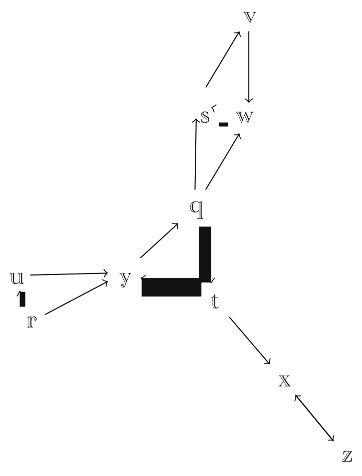
Remove lines coming from G, F isn't connected to anything:

A, B, D, E, C, H, G, F,

Remove F, I is pointing at J and J is the last one after I: A, B, D, E, C, H, G, F, I,

Problem 7: Strongly Connected Components (6 points)

Show how the Strongly Connected Components algorithm presented in class works on the graph below. Specifically, show the finishing times computed in line 1 and the forest produced in line 3. Assume that the DFS considers vertices in alphabetical order, and that the adjacency lists are in alphabetical order.



Q	13
R	1
S	12
T	14
U	2
V	10
W	9
X	15
Y	3
Z	17

R
U
Y
Q
S T
V X
W Z

Problem 8: Simplifying SCC. (4 points)

Professor Bacon claims that the algorithm for strongly connected components would be simpler if it used the original (instead of the transpose) graph in the second depth-first search and scanned the vertices in order of increasing finishing times. Does this simpler algorithm always produce correct results?

For a directed graph, probably not. Scanning the original doesn't take into account

Problem 9: Communication Networks (5 points)

One usage case for a graph is to use it to model communication networks. In all electronic communication networks, there is some probability that a message between node a and node b will fail— that is, the message won't successfully make it to node b.

Give an efficient algorithm that produces the most reliable path between two given vertices.

You are given a directed graph $G = (V, E)$. Each edge $(u, v) \in E$ has an associated value, $p(u, v)$ such that $0 \leq p(u, v) \leq 1$ and represents the probability of a successful transmission between u and v. Assume that all of these probabilities are independent.

Note: If event A is independent of event B, and event A happens with $p(A)$ probability, and event B happens with $p(B)$ probability, the probability that both event A and B happen is $p(A) \cdot p(B)$.

Give a short summary of your algorithm approach, psuedocode for the algorithm, and an estimate of the run time.

Question	Points	Score
General graphs	10	
DFS	3	
BFS	4	
White Hats, Black Hats	6	
Graph Arithmetic	4	
Topological ordering	5	
Strongly Connected Components	6	
Simplifying SCC.	4	
Communication Networks	5	
Total:	47	

SUBMISSION DETAILS

Things to submit:
• Submit your assignment in your Github repo.

- The written parts of this assignment as a .pdf named "CS5006_[lastname]_A4.pdf". For example, my file would be named "CS5006_Slaughter_A4.pdf". (There should be no brackets around your name).
- Make sure your name is in the document as well (e.g., written on the top of the first page).
- Make sure your assignment is in the A4 folder in your Github repo.

HELPFUL HINTS

- Ask clarification questions on Piazza.
- Remember, your write-up should convince graders and instructors that you are providing your own work and should showcase your understanding.
- Use the resources page on the course website for supplemental materials.
- In general, problems will be graded both on whether you are taking the right approach and whether you got the right answer. So, show your work and explain your thinking.